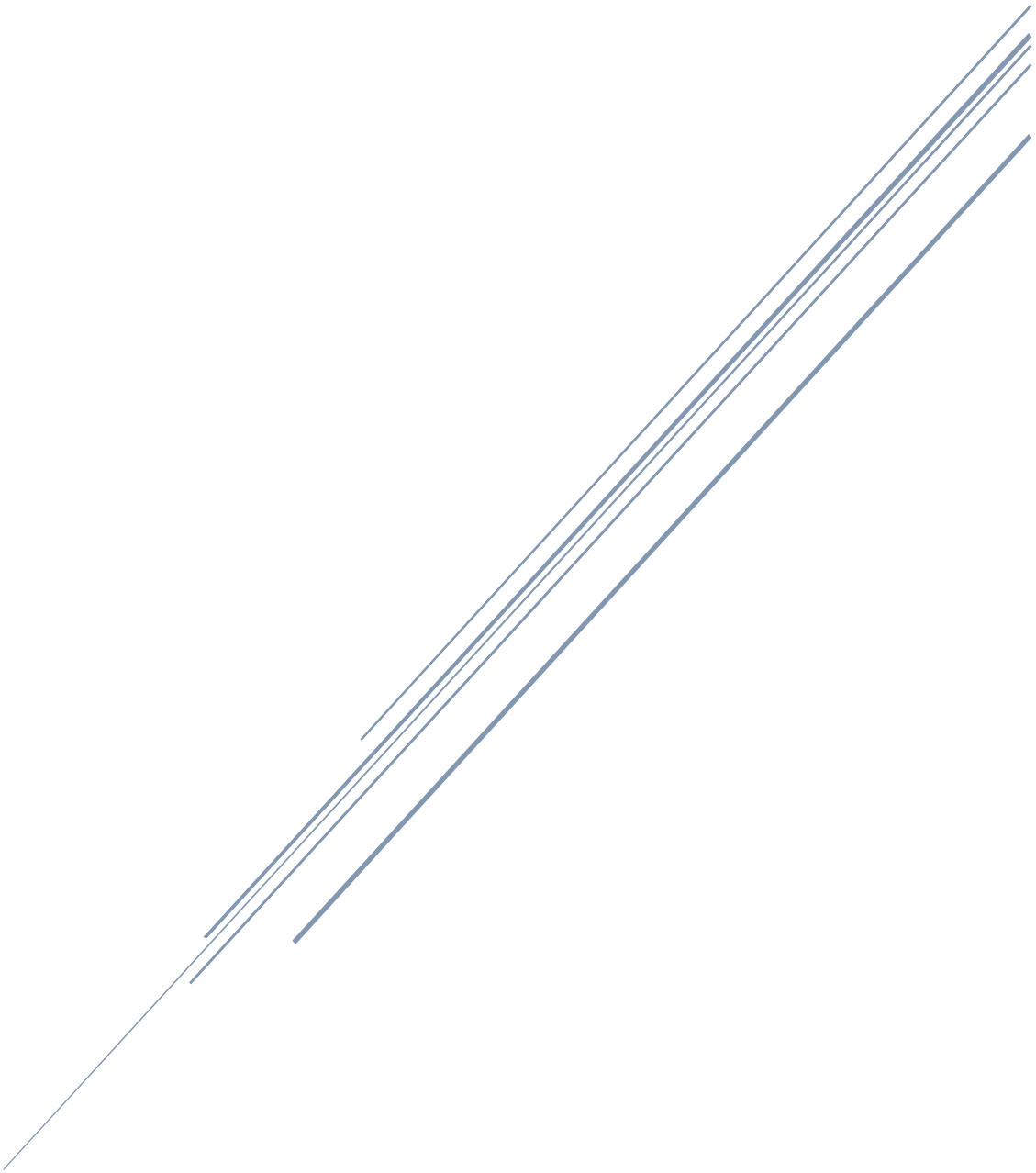
KÜNSTLICHE INTELLIGENZ DOKUMENTATION

Marc Pfitzenmaier, Björn Bauer (9973491)



DHBW
Künstliche Intelligenz

DHBW
Duale Hochschule
Baden-Württemberg

Inhalt

1. Konzept

A.1 Annahmen für das Programm

- Der Spieler ist allwissend. Er sieht alle Spielsteine (das gesamte Labyrinth) und den Spielstein außerhalb.
- Das Bewegen der Figur ist optional, es kann demnach mehrmals hintereinander das Labyrinth verschoben werden.
- Die Position des Einschubs des nicht genutzten Spielsteins wird durch den Spieler bestimmt
    - Der Spielstein kann sich nicht drehen
- Die Spielfigur wird, wenn sie rausfliegt auf das Feld gesetzt, dass sie hinausgeschoben hat (das Feld, das hineingeschoben wurde)

A.2 Einlesen der Daten (Labyrinth, Start, Ziel, freies Teil)

Bei Start des Programms müssen die oben genannten Daten eingelesen werden. Das Format dabei ist folgendes:

- Labyrinth
    - Wird in Form einer JSON Datei eingelesen in der 5*4 Felder beschrieben sind (separiert durch, in Form eines Array).
- Start/Ziel
    - `"startingPosition": { "row": 6, "column": 1 }`
    - `"goalPosition": { "row": 0, "column": 2 }`
- Freies Teil
    - `"freeTile": 2`
- `Positionen bei denen ein freies Teil eingefügt werden kann`
    - `"rowsWithTileInput": { "left": [1, 3, 5], "right": [2, 4] }`

A.3 Labyrinth Bauteile

Es handelt sich um ein 5*4 Felder großes Labyrinth, das aus 10 verschiedenen Bausteinen bestehen kann. Diese Bausteine werden, wie Bild 1 zeigt codiert

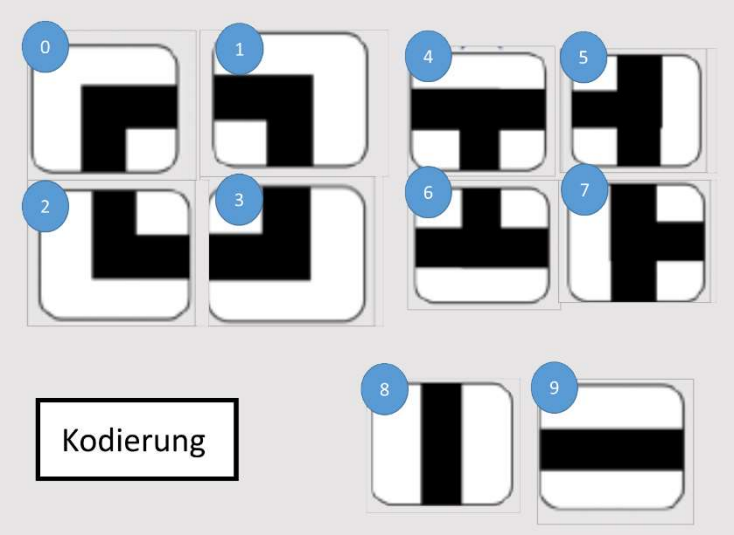


Abbildung 1: Codierung Labyrinth Bausteine

Jedes Labyrinth-Teil benötigt zusätzlich Regeln, in Welche Richtung gegangen werden kann und welches Bauteil an dieser Stelle möglich ist. Ein Beispiel hierfür wäre das bei Bauteil 0 nur nach rechts gegangen werden kann und auch nur, wenn sich hier das Bauteil 1, 4, 5, 6,3 oder 9 befindet. Außerdem kann nur nach unten gegangen werden, wenn sich hier Bauteil 2, 3, 8, 6, 5 oder 7 befindet. Ähnliche Regeln müssen für alle 10 Bauteile aufgestellt werden.

Code: (ergänzen und Formatierung anpassen)

```
class LabyrinthTile:
    """
    defines from which directions the labyrinth tile is accessible
    """

    def __init__(self, left_access: bool, top_access: bool, right_access:bool, bottom_access:
bool):
        """
        Creates a Labyrinth Tiles and defines from which direction it is accessible
        """
        self.left_accessible = left_access
        self.top_accessible = top_access
        self.right_accessible = right_access
        self.bottom_accessible = bottom_access


LABYRINTH_FILE_DICTIONARY
= {
                                    "0": LabyrinthTile(False, False, True, True),
                                    "1": LabyrinthTile(True, False, False, True),
                                    "2": LabyrinthTile(False, True, True, False),
                                    "3": LabyrinthTile(True, True, False, False),
                                    "4": LabyrinthTile(True, False, True, True),
                                    "5": LabyrinthTile(True, True, False, True),
                                    "6": LabyrinthTile(True, True, True, False),
                                    "7": LabyrinthTile(False, True, True, True),
                                    "8": LabyrinthTile(False, True, False, True),
                                    "9": LabyrinthTile(True, False, True, False),
                                    "10": LabyrinthTile(True, True, True, True),
                                    "11": LabyrinthTile(False, False, False, False)
                                }

"""
        Maps LabyrinthTile code to correct LabyrinthTile object
        - First Boolean Parameter: Access to the left side of the tile
        - Second Boolean Parameter: Access to the top side of the tile
        - Third Boolean Parameter: Access to the right side of the tile
```

- Fourth Boolean Parameter: Access to the bottom side of the tile

For Exam

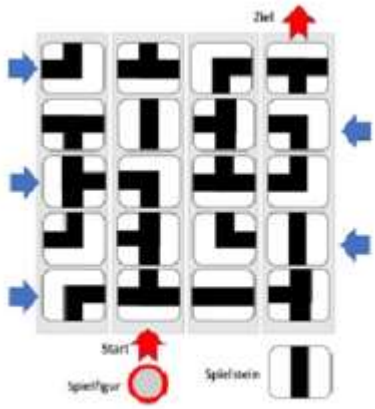
A.4 Verschieben Spielfigur (A*)

Die Spielfigur kann nur verschoben werden, wenn es einen weg gibt auf dem Sie Laufen kann (in Schwarz). Dies betrifft auch Start und Ziel.

Code:

A.5 Verschieben Labyrinth (A*)

Das Labyrinth kann zu jeder Zeit verschoben werden, es ist nicht nötig, dass sich die Spielfigur zwischen dem Verschieben von Bauteilen bewegt. Spielsteine können nur an 5 spezifizierten Punkten eingesetzt werden (in Abbildung xxx durch blaue Pfeile markiert).



Code:

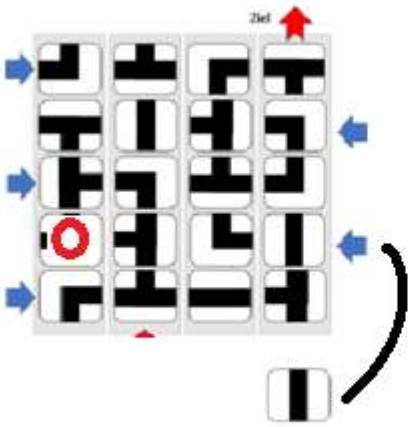
A.6 Spielzug

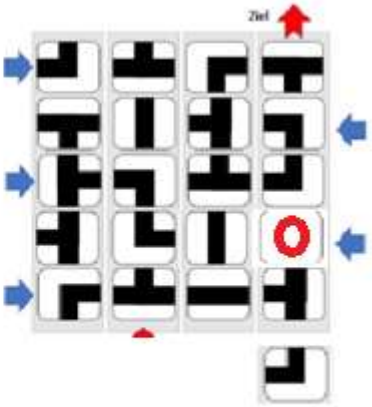
Ein Spielzug besteht aus dem Einsetzen eines Bauteils und dem anschließenden bewegen der Spielfigur. Die Spielfigur muss sich allerdings nicht bewegen, weshalb jedes Einsetzen eines Spielsteins als einzelner Spielzug anzusehen ist.

Code:

A.7 Spielfigur neu setzen

Wird die Spielfigur aus dem Spielfeld geschoben (durch das Einsetzen eines Spielsteins) so wird Sie auf den Spielstein, welcher hineingeschoben wurde, wieder eingesetzt. Dieses Verhalten wird in Bild xxx verdeutlicht.





Code:

Heuristik

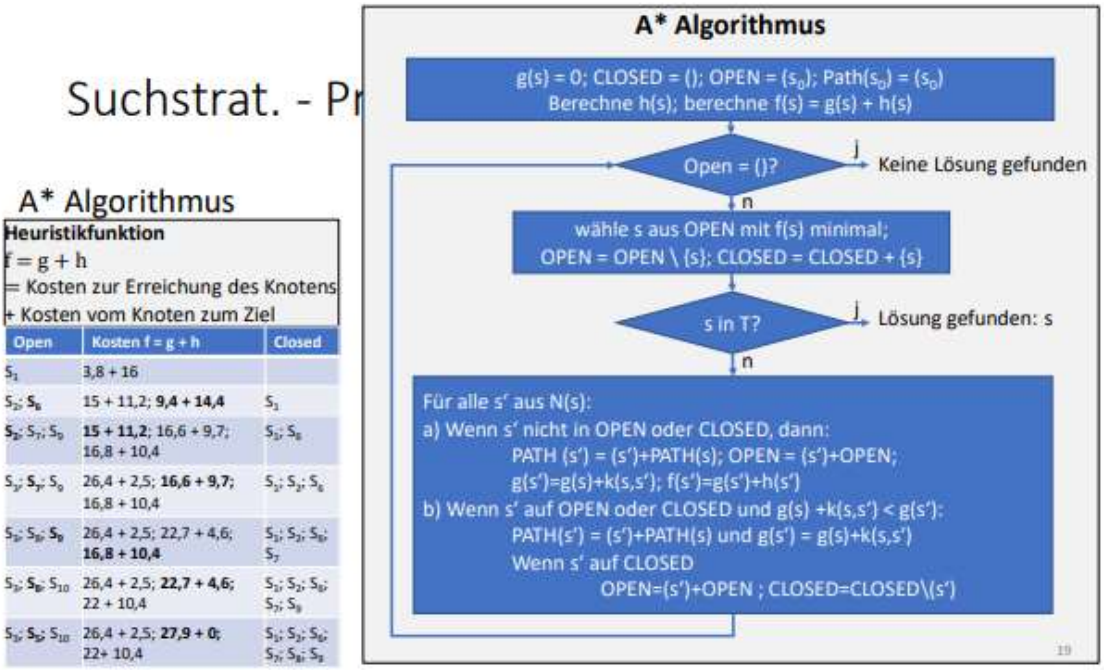
Eine genauere Definition, was Heuristik bedeutet finden Sie in Punkt 2. Bezug der Vorlesung. Für dieses Projekt wurde wie bereits erwähnt das Heuristische Suchverfahren A* gewählt. Hierfür setzt sich die Heuristik aus einer Kombination der Momentanen Distanz und der Durchschnittlichen Distanz zum Ziel zusammen. Die durchschnittliche Distanz wird hierbei durch die minimale Anzahl an Bausteinen die der Spielstein zum Ziel benötigt (ohne das er dem Weg folgt) angenähert.

Code:

```
def calculate_combined_heuristic_cost(self):
        """
        - Calculates combined heuristic cost by adding the current distance and an approximated distance to the goal
        - Approximates the distance to the goal by using the minimum number of tiles the player has to pass to reach the goal
        """
        self.combined_heuristic_cost = self.distance + abs(self.goal_column - self.player_column) + abs(self.goal_row - self.player_row)
```

2. Bezug zur Vorlesung

Bei dieser Aufgabe handelt es sich um die Implementierung des A*-Algorithmus (siehe Abbildung xxx), ein Heuristisches Suchverfahren (Vorlesung Intelligente Suchverfahren) zur Bestimmung des kürzesten Wegs von einem Starpunkt zu einem Zielpunkt. Besonderheit bei dieser Problemstellung ist, dass grundsätzlich zwei A* nötig sind, einen der den besten Weg für die Spielfigur findet und nach jedem Einsetzen des Spielsteins neu gestartet werden muss und einer, der den besten Punkt für das Einsetzen des Spielsteins bestimmt.



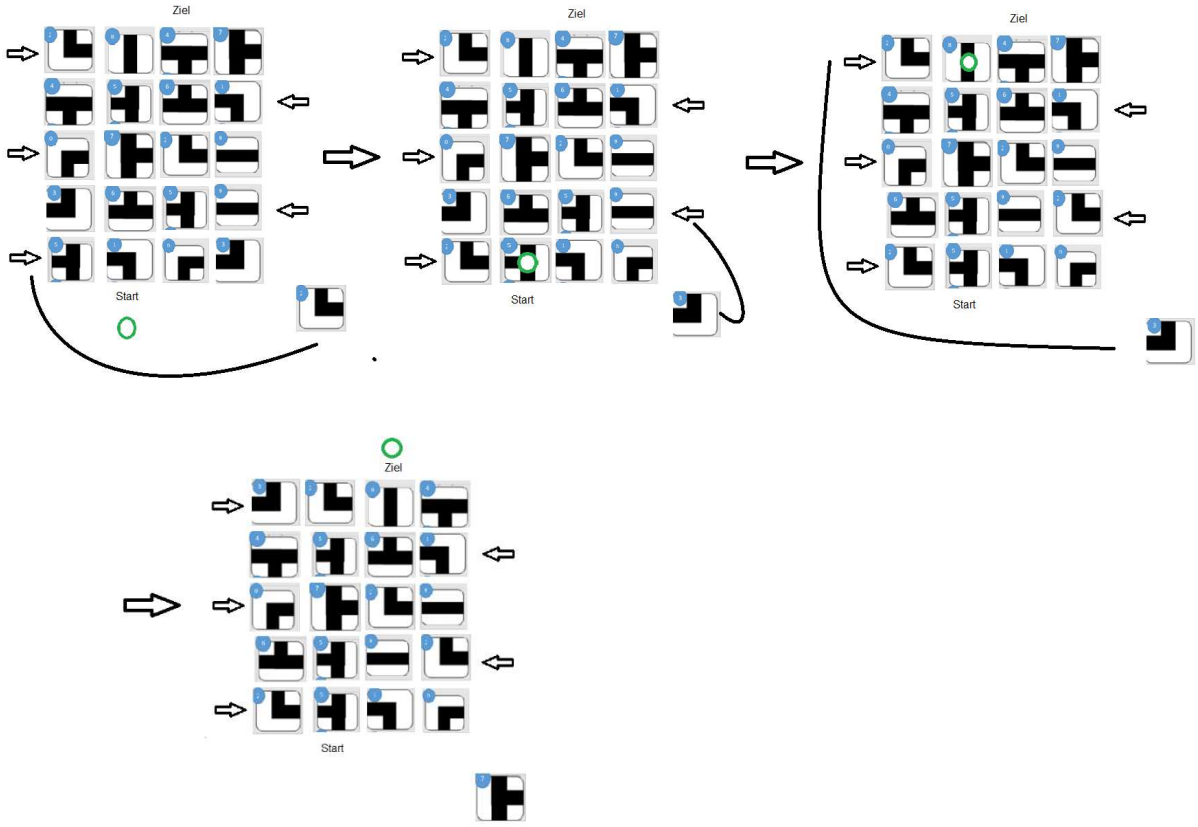
Eine der Besonderheiten dieses Algorithmus ist, dass bereits verworfene Wege unter bestimmten Umständen wieder verwendet werden können. Zu sahen ist dieses Verhalten in Abbildung xxx unter Regel b) Wenn s‘ auf closed.

Wie bereits angesprochen handelt es sich bei diesem Algorithmus um ein heuristisches Suchverfahren. In diesem Zusammenhang bezieht sich der Begriff der Heuristik auf eine Technik die entwickelt wurde um ein Problem schneller oder effektiver zu lösen insofern klassische Methoden zu langsam/ineffektiv sind.

3. Begründung des Entwurfs/der Umsetzung (Diskutieren der Konfiguration)

4. Test und Ergebnisbewertung (Diskutieren des Ergebnisses)

Das Labyrinth kann in 3 Zügen beendet werden (wie in Abbildung xxx gezeigt wird). Dabei muss vor jedem Laufen der Spielfigur ein Spielstein eingesetzt werden.



5. Quellen

Offene Punkte

- Lösungsqualität und Umfang der Funktionalität
- Korrekte Verwendung von Kernfunktionen
- Anpassung an die Aufgabenstellung